

Exercício 03

Observações Gerais

- I. A implementação da solução do problema proposto deverá:
 - a. Ser realizado em linguagem C/C++;
 - b. Ser feita no arquivo com o mesmo nome do exercício. Por exemplo: a solução do exercício 1 será realizada no arquivo `exercício_1.cpp`; do exercício 2 no arquivo `exercício_2.cpp`; e assim por diante;
- II. Todos os arquivos das implementações dos algoritmos de ordenação, lista linear simples, pilha, fila e árvore de pesquisa, necessários para a solução (conforme é descrito no enunciado) do exercício serão fornecidos;
- III. O aluno somente poderá realizar a implementação da solução no arquivo de trabalho do exercício (`exercício_1.cpp`, `exercício_2.cpp`, etc). Alterações nos outros arquivos não serão consideradas na correção;

Observações sobre a Compilação e Execução do Exercício

- I. O programa (solução do exercício) deverá compilar sem erros. Caso existam erros de compilação a solução não será considerada, ou seja, será atribuída nota igual a zero;
- II. O arquivo `exercício_x.cpp`, submetido e que contém a implementação da solução do exercício, será compilado com os arquivos idênticos ao anexo do enunciado do exercício. Por exemplo, o anexo do enunciado contém os arquivos: *listalinearsimples.h*, *listalinearsimples.cpp*, *exercício_1.cpp*, *main_exercicio_1.cpp*. O arquivo *exercício_1.cpp* deverá conter a solução do problema proposto e consequentemente, deverá ser postado como a solução. Durante a correção, o arquivo *exercício_1.cpp* (enviado como solução) será compilado com os arquivos *listalinearsimples.h*, *listalinearsimples.cpp* e *main_exercicio_1.cpp* idênticos anexos ao enunciado. Por isso, não altere os arquivos que não contenham a solução do exercício;
- III. Uma vez que o programa compile sem erros, ele não poderá apresentar advertências de compilação. Caso existam advertências de compilação, a nota final do exercício será reduzida em 20%;
- IV. O programa executável, gerado pela compilação, deverá iniciar sem cancelar sua execução (sem erros de execução SEGMENTATION FAULT, CORE DUMP, entre outros). Caso o programa executável cancele sua execução, a nota final atribuída ao exercício será zero.

Observações sobre o Teste da Solução do Exercício

- I. O teste será realizado em duas etapas:
 - a. A primeira etapa será um teste realizado com os mesmos dados constantes no enunciado do exercício;
 - b. A segunda etapa será um teste realizado com outros dados contendo as mesmas características dos tipos de dados do teste da primeira etapa, porém com uma quantidade de informações diferentes;

Observações sobre Plágios

- I. A verificação de plágio de soluções será realizada pela plataforma MOSS (<https://theory.stanford.edu/~aiken/moss/>);
- II. Caso sejam detectados mais do que 80% de similaridade entre as soluções dos alunos, será considerado um plágio e consequentemente a solução não será considerada, ou seja, será atribuída nota igual a zero;

Enunciado

No terceiro exercício você deverá criar uma fila (QUEUE) que armazene valores entre 0 e 100. Estes valores representam a idade de pessoas. O programa deverá, a partir dos dados armazenados na fila, calcular a distribuição de frequência das idades. A faixa de frequências deve ser a seguinte:

0 a 9, 10 a 19, 20 a 29, 30 a 39, 40 a 49, 50 a 59, 60 a 69, 70 a 79, 80 a 89, 90 a 100.

Observe o exemplo a seguir:

Idades inseridas na fila:

30, 31, 30, 29, 70, 10, 9, 8, 7, 55, 57, 72

Resultado esperado:

0	A	9	=	3
10	A	19	=	1
20	A	29	=	1
30	A	39	=	3
40	A	49	=	0
50	A	59	=	2
60	A	69	=	0
70	A	79	=	2
80	A	89	=	0
90	A	100	=	0

Arquivos anexos ao enunciado:

descriptor.h

descriptor.cpp

exercicio_3.h

queue.h

queue.cpp

exercicio_3.cpp

exercicio_3_main.cpp

OBSERVAÇÃO:

Você deverá implementar a seguinte função no arquivo `exercício_3.cpp`:

- `void showFrequencyDistribution(Queue * queue)`

Apenas o arquivo `exercício_3.cpp` deverá ser postado, via Teams, como a solução do exercício.